

REMARKS

Claims 1-15 were pending as of the action mailed on February 8, 2008. Claims 1, 9, 12 and 15 are in independent form.

Reexamination of the application and reconsideration of the action are respectfully requested in light of the foregoing amendments and the following remarks.

Section 103

Claims 1, 3-12, 14-15 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over U.S. Publication No. 2002/0083073 ("Vaidya") in light of U.S. Patent No. 6,675,230 ("Lewallen").

Vaidya is not relevant art.

Vaidya describes systems and methods for use by users in "managing a layered hierarchical data set." [Vaidya Abstract] "Embodiments of the invention are for managing a layered hierarchical data set." [Vaidya para. 44] "When a user accesses data element in the hierarchical data set, a sub-tree of the hierarchical data is presented to the user. When the user operates on a data element, the state attribute of the node associated with the data element is updated. Using the updated state attribute, the entire hierarchical data set is managed so that when a user accesses the data again, it is presented correctly." [Vaidya Abstract, para. 29]

It is important to recognize that Vaidya relates to the management of data that resides in a server and can be managed by multiple users. In fact, the hierarchical data set that is being managed typically resides in a backend database:

With reference to embodiments of the present invention, hierarchically organized information 950 typically resides in the back tier in a database 955, while the data is typically accessed by client computer 901. In operation, client computer 901 makes a request 960, to the database 955 for the hierarchically organized information 950. The request 960 is then used to present the user with an appropriate tree by accessing the database 955 and locating the proper nodes in the hierarchical information 960 to re-create a sub-tree 965 on the client computer 901. [Vaidya para. 93]

A user can access and operate on data in the layered hierarchical data set. When the user "accesses data in the hierarchical data set, a subset of the hierarchical data set (i.e., a sub-tree) is

presented to the user.” If the user does operate on the data, the data and its state attributes are modified. [Vaidya para. 47] “In operation, client computer makes a request, to the database for the hierarchically organized information. The request is then used to present the user with an appropriate tree by accessing the database and locating the proper nodes in the hierarchical information to re-create a sub-tree on the client computer.” [Vaidya para. 93; reference numbers omitted]

The hierarchical data set can be “cached in a server as a DOM tree.” [Vaidya para. 65, FIG. 2; emphasis added] It is important to recognize that such a DOM is similar in name only to the DOM built and used by a Web browser in a client, to which the present application relates.

“Typically, when a user asks for data, such as that found in a server with a hierarchical layout like that shown in FIG. 2 [the data cached in a server as a DOM tree], the user receives a merged view of the data . . . which is what a user would operate on” [Vaidya para. 67, *see also* para. 93 quoted above]

“When an user asks for the data, the user receives a merged tree view of the data. Therefore, the server has the responsibility of converging data into a merged tree, and converting the merged tree transactions to the actual tree on the server.” [Vaidya para. 33]

Vaidya describes an updateNode function that performs this responsibility of converging data. The updateNode function “converts the merged tree transactions to the actual tree on the server.” [Vaidya para. 70; emphasis added] The updateNode function is a function that can reside on a server. [Vaidya para. 70] It performs a number of functions on the user’s sub-tree, i.e., on the merged view of the hierarchical data set that the user computer received when the user asked for data. [Vaidya paras. 67, 0078, 0093]

Claim 1. To support the rejection of claim 1, the action asserts that:

Vaidya teaches . . . :

- Identifying in a web browser on a client computer a UI element being an element of a user interface page received from a server computer, rendered by the browser, and provided as a screen presentation to a user of the client computer, (See Para 93). . . . [Emphasis in original]

The applicant respectfully disagrees. Vaidya does not describe or relate to UI elements in a web browser client. In fact, the paragraph relied on in the action makes clear that exactly the opposite of what the applicant is claiming is what is described in Vaidya. That paragraph reads in its entirety as follows:

[0093] With reference to embodiments of the present invention, hierarchically organized information 950 typically resides in the back tier in a database 955, while the data is typically accessed by client computer 901. In operation, client computer 901 makes a request 960, to the database 955 for the hierarchically organized information 950. The request 960 is then used to present the user with an appropriate tree by accessing the database 955 and locating the proper nodes in the hierarchical information 960 to re-create a sub-tree 965 on the client computer 901.

This passage describes a process that finds the location of proper nodes in a server. It does not describe what is recited in the claim. In particular, it does not describe identifying anything in a web browser, as recited in the claim.

In addition, it does not describe identifying an element in a user interface page, which is also recited in the claim.

Further, it does not describe identifying anything in a user interface page that was received from a server, which is also recited in the claim.

To support the rejection of claim 1, the action further asserts that:

Vaidya teaches . . . :

- Identifying . . . a change of a user interface (UI) element that references a node of a document object model (DOM) hierarchy (Vaidya Para 0047-0048 and Figure 1 and Para 67) Vaidya teaches that a user operation on a DOM tree node is identified. . . .

The applicant respectfully disagrees. Vaidya does not describe or relate to UI elements that reference a DOM in the client.

The actual text that the action relies on, paragraphs 47, 48, and 67 of Vaidya, reads as follows:

[0047] Referring again to FIG. 1, at operation 110 a user accesses data in the hierarchical data set. In response, a subset of the hierarchical data set (i.e., a sub-tree) is presented to the user at operation 120. Next, it is

determined if the user has operated on a subset of the data at operation 130. If not, the process repeats at operation 120. Otherwise, the data and state attributes are modified at operation 140.

[0048] When a user again needs access to the hierarchical information, one embodiment of the present invention shown in FIG. 7 presents it to the user in the following manner. At operation 700, the state attributes in the tree are traversed so that the appropriate information (i.e., nodes) may be presented to the user. Next, at operation 710 it is determined whether the present state attribute has a default value. If the state attribute does in fact have a default value then it has not been changed. Thus, the process repeats and other state attribute are examined at operations 700 and 710 to find one that is non-default.

....

[0067] Typically, when a user asks for data, such as that found in a server with a hierarchical layout like that shown in FIG. 2, the user receives a merged view of the data. FIG. 3 is a merged view of the tree in FIG. 2 which is what a user would operate on when they access Star application 250. FIG. 3 is designed to show how a user might wish to change a preference in their layer and how this change should override the value in a node in the group or system layer.

Vaidya describes user operations on a DOM tree – more accurately, on a merged partial tree presented to a user –in paragraphs 65, 67 and 93. As noted above, the user operations of Vaidya do not relate to user interface elements, but rather to direct user manipulation of the hierarchical data set, cached on a server by a DOM tree, in those embodiments of Vaidya that use a DOM tree.

To support the rejection of claim 1, the action further states that:

Vaidya shows the user has identified the need to change information in the browser

The applicant does not see what disclosure in Vaidya this statement relates to, nor how this statement supports the rejection. If this is part of the basis for rejecting the claim, the applicant respectfully requests a supplemental action in which this is explained. A rejection violates 35 U.S.C. § 132 if it “is so uninformative that it prevents the applicant from recognizing

and seeking to counter the grounds for rejection.” *Chester v. Miller*, 906 F.2d 1574, 1578, 15 USPQ2d 1333, 1337 (Fed. Cir. 1990).

To support the rejection of claim 1, the action further asserts that:

Vaidya teaches . . . :

- Determining whether the change of the UI element can be applied to the DOM hierarchy by using an update function (Vaidya See Para 0069-0076)

The applicant respectfully disagrees.

Preliminarily, the claim actually recites, “determining in the browser whether the change of the UI element can be applied to the DOM hierarchy” The action does not purport to have addressed this feature. For this reason alone, the rejection should be withdrawn.

Substantively, the portion of Vaidya relied on in the action reads as follows:

[0069] updateNode

[0070] One embodiment of the present invention converts the merged tree transactions to the actual tree on the server. In one embodiment, the function responsible for this is called updateNode, which may reside on a server, such as a Registry server. This function performs a variety of operations on the client side. These operations include, for instance:

[0071] Override value: change the default value to a new overridden value, or change an already overridden value to a new overridden value.

[0072] Delete: remove a default node, or delete a user defined node (one that does not exist in the default tree).

[0073] Reset: reset the node to a default value.

[0074] Add: add a new node to the user's sub-tree.

[0075] Replace: replace a default node with a new node with either the same name, or different name to the user's sub-tree.

[0076] FIG. 4 is a partial view of the DOM tree of FIGS. 2 and 3 where updateNode function has been used to change the Font layer 400 from Times 292 (in both FIGS. 2 and 3) to Arial 405, and the Size node 410 under Lines 415 has been changed from 12 in FIG. 2 to 10 in FIG. 3 to 14 in FIG. 4.

Nothing in these paragraphs describes or suggests determining whether an update function can be used to make the identified change in the DOM. Nothing in Vaidya suggests anything other than that the described updateNode function can perform all the operations required of it; and no mention is made of any changes that can be made, but not made by an update function. The rejection under § 102 is therefore improper, because the action has failed to show that Vaidya discloses “determining whether,” as explicitly recited in the claim.

To support the rejection of claim 1, the action further asserts that:

Vaidya teaches . . . :

- If the change can be applied by using the update function, finding in the DOM hierarchy the node that is referenced by the UI element and modifying at least one attribute related to the node according to the change by using the update function (Vaidya Para 0076- 0077) Vaidya teaches that at least on attribute is changed through the update function for the node.

The applicant respectfully disagrees.

The portion of Vaidya relied on in the action reads as follows:

[0076] FIG. 4 is a partial view of the DOM tree of FIGS. 2 and 3 where updateNode function has been used to change the Font layer 400 from Times 292 (in both FIGS. 2 and 3) to Arial 405, and the Size node 410 under Lines 415 has been changed from 12 in FIG. 2 to 10 in FIG. 3 to 14 in FIG. 4.

[0077] FIG. 5 shows a table of how these operations are accomplished. The first column shows the state value for a node before the updateNode operation, while the first row shows the client actions taken. The other rows show the final state value of the node after the updateNode operation is successfully completed. According to another embodiment of the present invention, when each node undergoes an updateNode operation the server has to update the information in the cache, and has to relay the same information to the external storage media.

Initially, although the actions states, “If the change can be applied (emphasis added),” nothing in the cited disclosure teaches or suggests that the change cannot be applied. Consequently, the features recited in the claim that relate to dealing with this possibility are not found in the reference.

Next, the claim requires that the update function be applied in the browser. As explained above, the updateNode operations of Vaidya are applied in the server.

To support the rejection of claim 1, the action finally asserts that:

Vaidya teaches . . . :

- Else, setting a dirty flag. (See Para 0062) Vaidya teaches that the replaced state attribute for a given node was not known to the previous layers and is new and will have to be updated in the cache.

The applicant respectfully disagrees. Preliminarily, the action does not – and could not – state that Vaidya actually teaches setting a dirty flag, which is what the claim recites. For at least this reason, a rejection under § 102 is improper and should be withdrawn.

The teaching actually found in Vaidya is that the addition of a node into the hierarchical data set by a user causes that node to have the state “replaced.” In fact, all that the portion of Vaidya relied on in the action states is the following:

[0062] Replaced: indicates that this node is not defined in the previous layers. The owner of the current layer has created this node, and another node with the same name may exist in another layer, but is replaced by this new node.

Furthermore, the claim recites that setting the dirty flag occurs as an alternative to applying an update function, i.e., when no update function is applied. Contrary to that, in Vaidya the updateNode function is always applied and, in fact, it is this function that changes the state to “replaced.” [Vaidya para. 77 and FIG. 5] Therefore, the alternative operation recited in the claim is not disclosed by Vaidya, and the rejection under § 102 is improper.

Finally, the cache referred to in the action is a server cache, and the operations recited in the claim occur in the client.

The action does not rely on the second reference, Lewallen, for any teaching that the applicant has shown is not found in Vaidya. The applicant respectfully submits that Lewallen does not provide any such teaching. Thus, the *prima facie* obviousness of the claims has not been established, and the rejection under 35 U.S.C. § 103 should be withdrawn.

The action relies on Lewallen for some teachings that the action acknowledges are not found in Vaidya.

The conclusion of the analysis in the action appears to be the following two sentences on the bottom of page 4:

Lewallen and Vaidya teach processes of updating nodes in a document object model. They both teach processes of accessing a given node and element.

The applicant would not dispute that such techniques are generally in the state of the art. However, such general abstract statements are not what the claims actually recite. For a rejection to be proper, it must address the actual limitations of the claims, which the action has not done.

The action goes on to conclude that,

Accordingly, it would have been obvious to one of ordinary skill in the art at the time of the invention having the teachings of Vaidya and Lewallen in front of them to modify the system of Vaidya to include the client side manipulation of the node elements by storing a new element as a node in the DOM.

However, combining the teachings of Vaidya and Lewallen would not result in the claimed invention. First, as set for above, Vaidya lacks many of the features attributed to Vaidya in the action. Second, a client-side manipulation of node elements has no place in Vaidya, because Vaidya is a server-side technology. Third, the claim does not recite "storing a new element as a node in the DOM," as mentioned in the action; instead, it recites modifying node attributes. And more particularly, it recites modifying an attribute by "using at least one of the update functions of the delta renderer to directly modify the attribute."

For all of the foregoing reasons, the applicant respectfully submits that the rejection of claim 1 and its dependent claims should be withdrawn.

Claim 3. To support the rejection of claim 3, the action asserts that:

Vaidya teaches the product wherein the dirty flag indicates invalidity of a cached DOM representation of the UI element (See Para 0060)

The applicant respectfully disagrees. Paragraph 0060 reads in its entirety as follows:

[0060] Modified: indicates that the node in the current layer has a value different from the default value.

Nothing in this paragraph refers to a cached representation of part of a DOM, as recited in the claim. (Recall that, as explained above in reference to paragraph 0065, the entire DOM of Vaidya is itself a cache on a server of the hierarchical data set that is being managed by a user.)

Moreover, that a node in a tree has a value that differs from a default value does not mean that the node's value is invalid.

To support the rejection of claim 3, the action further asserts that:

Vaidya teaches the state of the node has changed and therefore the cache and the tree are not in sync and needs to be updated (See also Para 0057-0063).

The applicant respectfully disagrees.

First, as already mentioned, the cache in Vaidya is the DOM, so they are necessarily in sync. However, if by "the tree" the action is referring to the hierarchical data set that is being managed, then "the tree" of Vaidya is not in a Web browser in a client computer, as recited in the claim, and so does not correspond to anything recited in the claim.

Second, the paragraphs of Vaidya cited in the action do not support the conclusion that "the cache and the tree are not in sync and needs to be updated." They read as follows:

[0057] According to one or more embodiments of the present invention state attributes are added to each node in the hierarchy, for instance in the manner shown in FIG. 8. The state attribute has information relating to the last operation that was performed on its corresponding data element.

[0058] Possible values of the state attribute include:

[0059] Default: indicates that this node has come from the previous layer or the system default layer. This is the default value of the attribute.

[0060] Modified: indicates that the node in the current layer has a value different from the default value.

[0061] Deleted: indicates from the current layer's perspective that this node and all its sub-nodes are deleted (data does not exist).

[0062] Replaced: indicates that this node is not defined in the previous layers. The owner of the current layer has created this node, and another node with the same name may exist in another layer, but is replaced by this new node.

[0063] Depending on the value in the state attribute, different merged DOM trees are presented to the user. One example is shown in FIGS. 2-4.

It does not appear that anything in these paragraphs relates to data not being in sync and requiring an update.

For the foregoing additional reasons, the applicant respectfully submits that the rejection of claim 3 should be withdrawn.

Claim 4. To support the rejection of claim 4, the action first acknowledges that Vaidya does not teach the recited limitation:

Vaidya does not expressly teach the product wherein the dirty flag indicates invalidity of a cached DOM representation of a further UI element that is a parent of the UI element.

Then the action proceeds to assert:

However, this limitation would have been obvious . . . in view of Lewallen, because Lewallen teaches a process of checking to see if the node type is supported and if not then a new element is created and embedded in the DOM. The process in Lewallen teaches modifying the DOM for the purposes of changing the code statements to run on the native system where the objects were not previously recognized.

The applicant respectfully submits that this statement bears no relationship to what is actually recited in the claim.

For the foregoing additional reasons, the applicant respectfully submits that the rejection of claim 4 should be withdrawn.

Claims 7 and 8. To support the rejection of claims 7 and 8, the action asserts that:

With respect to dependent claims 7-8, as indicated in the above discussion, Vaidya in view of Lewallen teaches every element of claim 3 and 4.

The applicant respectfully disagrees for the reasons set forth above in reference to the rejections of claims 3 and 4.

For the foregoing additional reasons, the applicant respectfully submits that the rejection of claims 7-8 should be withdrawn.

To support the rejection of claims 7 and 8, the action further asserts that:

Vaidya teaches access a sub-tree of a data set that is stored as a DOM tree that is rendered on the client, A DOM tree is a hierarchical tree with UI elements. (See Para 0047-0048 and 0073-0075). Vaidya teaches accessing the tree to change the font element of a given node, which is an example of accessing the sub-tree of the DOM tree.

The applicant respectfully disagrees.

As explained earlier, user operations on a DOM tree do not relate to user interface elements, but rather to direct user manipulation of a hierarchical data set, which is cached on a server by a DOM tree, in those embodiments of Vaidya that use a DOM tree.

Also, as noted above, paragraphs 73-75, which are reproduced below, do not teach a cached sub-hierarchy in a client computer that is a cached representation of a UI element that has experienced a change that was identified in a Web browser in the client computer, as recited in the claim. All these paragraphs describe is the addition, replacement, or resetting of a node to a tree:

[0073] Reset: reset the node to a default value.

[0074] Add: add a new node to the user's sub-tree.

[0075] Replace: replace a default node with a new node with either the same name, or different name to the user's sub-tree.

For the foregoing additional reasons, the applicant respectfully submits that the rejection of claims 7 and 8 should be withdrawn.

Claim 9. To support the rejection of claim 9, the action asserts that:

Vaidya teaches . . . :

receiving a request to render a user interface (UI) element of a UI-tree UI tree in a client computer, the UI tree representing in the client computer a hierarchical structure of UI elements contained in a user interface page received in the client computer from a server computer, the UI element being an element of the user interface page (See Para 33). Vaidya teaches the user requests to receive a UI interface that is a DOM tree. The DOM tree contains user interface elements and the DOM is received at the client from the server. [emphasis in original]

The applicant respectfully disagrees.

The part of Vaidya on which the action relies is reproduced below in its entirety:

[0033] In order to keep track of data modified by various clients, another embodiment of the present invention has the data as a single Document Object Model (DOM) tree. When an user asks for the data, the user receives a merged tree view of the data. Therefore, the server has the responsibility of converging data into a merged tree, and converting the merged tree transactions to the actual tree on the server. This is accomplished by examining the data elements in the hierarchy where the associated state attribute is no longer in a default state and presenting the data elements associated with the non-default state attribute and their parent nodes, continuing up to the root of the tree. Then, merging any changes made on the sub-tree back to the complete tree by updating the state elements.

This paragraph simply does not state what the action says it does. What the user receives is data in the form of a merged tree. There is no mention of a user interface page, which is a feature expressly recited in the claim.

To support the rejection of claim 9, the action further asserts that:

Vaidya teaches the system renders the UI tree (See Figure 2).

The applicant respectfully disagrees.

Vaidya does not teach rendering a user interface element of a user interface page that has a hierarchical structure represented by the recited UI tree in the client computer.

Figure 2 of Vaidya is described as follows:

[0036] FIG. 2 is a view of a DOM tree suitable for use with the present invention.

[0036] FIG. 2 shows data cached in a server as a DOM tree. . . .

Figure 2 does not teach the rendering of a UI tree.

For the foregoing reasons, the applicant respectfully submits that the rejection of claim 9 should be withdrawn.

To support the rejection of claim 9, the action finds in Lewallen the limitations related to determining whether a dirty flag is set. These grounds for rejection have already been addressed in reference to claims 1, 3, and 4.

For the reasons set forth above in reference to claims 1, 3, and 4, and for the reasons just set forth in reference to claim 9, the applicant respectfully submits that the rejection of claim 9 and its dependent claims should be withdrawn.

Claims 12 and 14. Claims 12 and 14 were rejected “along the same rationale” as claims 1 and 3.

For the reasons set forth above in reference to claims 1 and 3, the applicant respectfully submits that the rejection of claims 12 and 14 should be withdrawn.

Claim 15. Independent claim 15 was rejected “along the same rationale” as claim 9.

For the reasons set forth above in reference to claim 9, the applicant respectfully submits that the rejection of claim 15 should be withdrawn.

Remaining claims. The remaining claims depend from the claims that have been addressed, and the rejection of the remaining claims should be withdrawn for at least the reasons set forth in reference to the respective independent claims or corresponding dependent claims.

Conclusion

For the foregoing reasons, the applicant submits that all the claims are in condition for allowance.

By responding in the foregoing remarks only to particular positions taken in the action, the applicant does not acquiesce with other positions that have not been explicitly addressed. In addition, the applicant's selecting some particular arguments for the patentability of a claim should not be understood as implying that no other reasons for the patentability of that claim

Applicant : Andreas Roessler
Serial No. : 10/677,121
Filed : September 30, 2003
Page : 19 of 19

Attorney's Docket No.: 13913-058001 / 2002P00218
US01

exist. Finally, the applicant's decision to amend or cancel any claim should not be understood as implying that the applicant agrees with any positions taken in the action with respect to that claim or other claims.

Please apply any other charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: 06 May 08

/Hans R. Troesch/

Hans R. Troesch
Reg. No. 36,950

Customer No. 32864
Fish & Richardson P.C.
Telephone: (650) 839-5070
Facsimile: (650) 839-5071